variable manual page - Tcl Built-In Commands

NAME

variable — create and initialize a namespace variable

SYNOPSIS

variable *name*
variable ?*name value...*?

DESCRIPTION

This command is normally used within a **namespace eval** command to create one or more variables within a namespace. Each variable *name* is initialized with *value*. The *value* for the last variable is optional.
If a variable *name* does not exist, it is created. In this case, if *value* is specified, it is assigned to the newly created variable. If no *value* is specified, the new variable is left undefined. If the variable already exists, it is set to *value* if *value* is specified or left unchanged if no *value* is given. Normally, *name* is unqualified (does not include the names of any containing namespaces), and the variable is created in the current namespace. If *name* includes any namespace qualifiers, the variable is created in the specified namespace. If the variable is not defined, it will be visible to the **namespace which** command, but not to the **info exists** command.

If the **variable** command is executed inside a Tcl procedure, it creates local variables linked to the corresponding namespace variables (and therefore these variables are listed by **info vars**.) In this way the **variable** command resembles the **global** command, although the **global** command resolves variable names with respect to the global namespace instead of the current namespace of the procedure. If any *value*s are given, they are used to modify the values of the associated namespace variables. If a namespace variable does not exist, it is created and optionally initialized.

A *name* argument cannot reference an element within an array. Instead, *name* should reference the entire array, and the initialization *value* should be left off. After the variable has been declared, elements within the array can be set using ordinary **set** or **array** commands.

EXAMPLES

Create a variable in a namespace:

```
namespace eval foo {
    variable bar 12345
}
```

Create an array in a namespace:

```
namespace eval someNS {
    variable someAry
    array set someAry {
        someName  someValue
        otherName otherValue
    }
}
```

Access variables in namespaces from a procedure:

```
namespace eval foo {
    proc spong {} {
        # Variable in this namespace
        variable bar
        puts "bar is $bar"

        # Variable in another namespace
        variable ::someNS::someAry
        parray someAry
    }
}
```

SEE ALSO

global, **namespace**, **upvar**